Fermi National Accelerator Laboratory

FERMILAB-Conf-92/13

High Performance Parallel Local Memory Computing at Fermilab

T. Nash
Computing Division
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

January 1992

Published in the Proceedings of *WHP92 on Heterogeneous Processing*, Beverly Hills, California, March 23, 1992.

Disclaimer

High Performance Parallel Local Memory Computing at Fermilab

Presented by Thomas Nash*

Computing Division
Fermi National Accelerator Laboratory, Batavia IL 60510

Introduction

Both experimental and theoretical approaches to high energy physics (HEP) at Fermilab are computer technology limited: no one can identify a requirement on computing or data capacity that is independent of cost or other realities. Fermilab has been forced, therefore, to turn significant attention and resources to finding extremely cost effective solutions to its computing using whatever technology is available, and in the process has become a pioneer in goal-driven computer science, integrating commercial solutions at the chip, board, and system level

The most effective HEP computing solutions are general purpose parallel systems which take advantage of the scientist's ability and willingness to identify explicitly the structure of the problem. Science, almost by definition, deals with regular problems which intrinsically offer parallel solution. It is straightforward to recognize that a lattice problem maps obviously to a grid of processors. Similarly, it is almost intuitive to recognize that the independent events resulting from the collision of particles in a HEP experiment can be dealt, one at a time, to the individual nodes of an event oriented parallel farm of computers. This explicit parallelism is the disting-uishing feature of the advances in computing developed in the HEP context. In recent years, most obviously first in the hypercube movement, and now in the surge of interest in networked parallel workstation "clusters", explicit parallelism has become a recognized force in the computer science attack on the problems of parallel computing.

Early work on parallel processing led directly to the two main explicitly parallel, local memory computing directions that dominate Fermilab's high performance computing today: farms of RISC technology workstations for experiment triggering and reconstruction, and the ACPMAPS massively parallel computer that supports theoretical calculations. Likewise, early efforts in software tools which allow the scientist to focus on the problem not the architecture have evolved into the modern packages CPS and Canopy which, respectively, are used in the contexts of networked farm systems and closely coupled massively parallel systems.

RISC processor farms and CPS

Today there are over 400 nodes in farms at Fermilab. A farm is a collection of loosely coupled processors that are managed as a single system. The most recent Fermilab farms used in experiment event reconstruction consist of two commercial species of RISC workstation processor nodes: Silicon Graphics, Inc. (SGI) and IBM RS/6000.

The processor farms

The processors in a farm are connected over a communications network so they are able to exchange data. That can be VME, Ethernet, or whatever is practical. Most activity is now via TCP/IP. This allows all the computers to share data from a single set of tape and disk drives. The sharing of these peripheral devices and the fact that a farm works collectively on a single problem at a time distinguishes farms from mere collections of workstations.

Each farm is composed of approximately 100 processors, serving as worker nodes, tape server nodes, and as NFS server nodes. Each runs its vended version of a UNIX operating system. The CPUs are rack mounted in tiers and each has at least 12 MB memory, 160 MB disk and Ethernet capability. See Table for other information. The farms are configured into "farmlets" of approximately 12 - 25 processors, and this subunit is on one Ethernet segment, separated by an Ethernet bridge or router. Each "farmlet" contains an NFS server, a tape server, and the balance are used as worker nodes. Local disk on each node has a copy of the operating system and paging space. The worker nodes all remote mount the Fermilab local executables from their "farmlet" NFS server, as well as scratch space areas for the batch jobs. Communication outside of the "farmlet" is generally limited to user logins and connections to the CPS Job Manager, which oversees the job.

* This report summarizes the efforts of many individuals in the Computing Division at Fermilab. Thanks to E. Schermerhorn, M. Fausey, M. Fischler, R. Pordes, S. Wolbers, for preparing this report.

Farms at Fermilab

IBM RS/6000	SGI, Inc.
Worker nodes	
63 Model 320s	22 Model 4D25s
36 Model 320Hs	78 Model 4D35s
Server nodes	
3 Model 530	3 Model 4D25s
2 Model 320s	3 Model 4D310s
Operating System	
AIX 3.1.5	IRIX 3.3.2
	IRIX 3.3.3

The software support

The software environment is layered. This approach utilizes the features of the UNIX operating system, and each farm implements its own vendor's version of UNIX. Lying above that is the communications and networking layer. This contains TCP/IP and NFS at a minimum. Topmost is special software which has been written at Fermilab to make it possible for scientists to use a farm effectively in a manner similar to that of a single large computer. Called "Cooperative Processes Software" (CPS), this software runs on a variety of computer platforms, and is designed to accommodate heterogeneous farms made up of different types of computers.

CPS is a tools package which eases splitting a job across a set of processes. The processes may be distributed over a set of computers. From the standpoint of hardware the CPS environment is a set of networked UNIX workstations. However, this is not the way the scientist sees the CPS environment; the process oriented applications software environment the user encounters is a fully-connected group of processes, each having equal access to each other and peripheral devices. CPS supports general process interactions including:

data transfers; send data to another process, and get data from another process.
remote subroutine calls; a process calls a subroutine on another process.
process queues and synchronization mechanisms; determine process availability for data or remote call, and coordination of transfers and calls.

An attractive feature of CPS for many is that it can be used with or without explicit message passing. While message passing can be specified for interprocess communication, CPS organizes process execution by queuing and dequeuing from, for example, READY and DONE queues. The user may define his own set of queues. Message passing underlies this scheme, as well as the CPS subroutine calls, but it remains hidden from the user unless it is explicitly used.

Important CPS subroutines

data transfers
- acp_declare_block
  declare array as possible source/destination of data transfer
- acp_send
  send data to a block in another process
- acp_get
  get data from a block in another process

synchronization and queues
- acp_declare_queue
  declare process eligible for a particular queue
- acp_queue_process
  place a process ID on a queue
- acp_dequeue_process
  remove a process ID from a queue
- acp_wait_queue
  wait for a queue to become full or empty
- acp_sync
  wait for set of processes to all arrive at synch point

remote subroutine calls
- acp_declare_subroutine
  declare subroutine eligible to call from another process
- acp_service_calls
  declare process to be a dedicated remote server
- acp_call
  call a subroutine in another process

Program development with CPS begins by incorporating CPS subroutines into the application code using the tool kit of explicit parallelism directives One then compiles the program on target computer(s) and links the program on the target computer(s) with the CPS system library. Applications debugging is simplified because jobs operate identically under CPS on single or multiple computers. The job manager starts the set of processes running, manages the interactions between the processes, and shuts down the processes at the end of the job.

CPS has a batch facility that supports production jobs. Users submit CPS jobs to UNIX batch queues, one queue for each processor farm or production system, depending on how the complex is configured. The next job on the queue is started when a farm becomes free. The various phases in the Batch sequence - ondeck; copy; execution; and, copyback - are pipelined, allowing multiple jobs to be handled. CPS batch supports computer center operations with an operator console that displays tape mount requests and job status information.

To date, the CPS software system has been ported to a large number of hardware platforms. Current implementations at FNAL assume all processes from a job are executing on the same computer platform, running identical environments. However, the CPS primitives

were originally designed to be platform independent, and thus offer the opportunity to manage distributed heterogeneous parallel processing. The current implementation of CPS (V2.5) can handle heterogeneous farms as has been demonstrated at a sister laboratory at another site.

We are planning a project to integrate CPS with a second Fermilab software system, UNIX Product Support (UPS) to provide a consistent automatically uniform environment across the varied platforms of heterogeneous networked system.

Creating consistent environments

As it exists now, UPS manages software packages across a set of different computer platforms, but has no functionality for job execution. Its appeal is that it simplifies and organizes the task of using a variety of computing platforms to work on the same application. It handles the housekeeping associated with developing and using applications, especially over long periods of time, in multiple computing environments.

UPS utilities provide uniform mechanisms for defining the versions of software to be used; user access to the required environmental variables and other parameters needed in order to use the software; distribution of selected software to remote computers; generation of the final application (files such as executables, data files etc - associated with the job to be run); provision of a platform independent process operating environment; census of the software installed on remote computers; archival and retrieval of old versions of software; etc.

The UPS kernel supports multiple simple databases of lists of software programs with associated attributes of version, platform type, proprietary nature, and dependency of the software on other software packages. A single UPS data base can be used by multiple, different computers, or multiple data bases can be used on a single computer. UPS provides utilities for handling the information contained in these databases, and for managing the software so described.

The extensions to CPS/UPS we plan would provide the framework to support the development, building, and execution of a multiprocess application where the functions to be performed are distributed among several different platforms of UNIX computers (IBM RS/6000, SGI, DEC5000 and Sparcstation IIs) and specified portions of the data to be processed are transferred between the processes.

To support execution of the processes as a single job, the system must provide for distribution of the correct versions of the requisite files to the various computers; for the transparent transfer of data between platforms that may have different dataword byte ordering or floating point representations; for the coordinated startup, monitoring and stopping of the processes in a uniform manner across the different computer platforms; and for a uniform and centralized mechanism for the reporting of results at the application level. All the interprocess communication, synchronization, management and data transfer mechanisms must be computer hardware and software independent.

For example, in the domain of HEP, the system we propose would allow experiment event data to be read from I/O media on a Silicon Graphics, the core of the data processed on an IBM RS/6000, and the results of the physics analysis visualized on a Decstation 5000. To achieve this requires a system that allows the application developer to specify what source files are to be compiled and linked on which platform; what dependencies these compilations and links have on each other; and what data and configuration files are needed on which platform. The system will provide for the requisite files and their updated versions to be available on the specified computer at the appropriate time. It will maintain a history of the application generation for later checking as needed by the application user or maintainer.

ACPMAPS: Massively parallel computer and Canopy

Fermilab efforts to support theoretical computations in HEP led to the development of a massively parallel local memory computer that is ideally suited for the lattice gauge calculations of quantum chromodynamics, QCD, one of the biggest consumers of computer cycles nationwide. This highly parallel, cost effective computer is called the Advanced Computer Program Multi-Array Processor System, or ACPMAPS.

ACPMAPS is a multiprocessor computer with hundreds of individual processor modules that can work in parallel on a single problem. It uses an innovative scheme for communication between its constituent processors. The connectivity can be thought of as similar to a modern telephone switching network, but at much higher speeds. Each processor module resides in one of the sixteen slots of a Bus Switch Backplane (BSB) crossbar switch crate which takes on a role analogous to a local telephone exchange. The BSB allows a processor in one BSB slot to make a point-to-point connection with a processor in any other BSB slot. Once a connection has been established, the processor that initiated the connection can read and write the memory of the other processor at a rate of 20 megabytes per second. The time to establish a connection is only a few micro-seconds.

Up to eight pairs of processors may communicate simultaneously, thus providing an aggregate communication of 160 megabytes per second within the crate. An additional module, the Bus Switch Interface Board (BSIB), provides a mechanism for communicating between BSBs. A BSIB in one crate can be connected via twisted-pair cables to a BSIB in another crate. This provides a 20 megabyte per second link between the two BSB slots like a long distance call. The ACPMAPS computer makes extensive use of BSIB modules--approximately one half of all BSB slots are filled with

BSIBs. This patented switch hardware allows systems to be interconnected in a large variety of ways with specific configurations optimized for the type of calculation to be performed. In the Fermilab installation, targeted at lattice gauge, the system is set up as 4 fully connected 3 x 3 planes of crossbar crates.

ACPMAPS also contains a highly parallel I/O subsystem. Approximately one half of the BSBs have one of their slots connected to a VME backplane in the same physical crate which in turn provides a connection to a SCSI bus. Each SCSI bus contains two 676 megabyte disk drives and two Exabyte 8mm tape drives (up to two gigabytes of storage per tape).

ACPMAPS upgrade

A new ACPMAPS processor module containing two 40 MHz Intel 860 microprocessors.per board, has been designed and built to upgrade the ACPMAPS processing power. This processor is compatible with the same BSB/BSIB communications mechanism. The original processing modules are being removed and replaced by the new modules. The commitment not to interrupt ongoing production physics dictates a cautious pace for this replacement process. Existing physics applications programs (that had already been written for ACPMAPS) run on the new modules with no changes. The underlying C software required only minor adaptations for the new processor, and a library of optimized computational kernels is being ported to the i860 chip architecture. These software changes represent a moderate systems effort, and are invisible to the ACPMAPS user community. The upgraded ACPMAPS will have ten times the computing power of the original ACPMAPS--the ability to perform at a peak rate of 50 GFLOPS.

Summary of upgrade

Existing ACPMAPS	Upgraded ACPMAPS
Aggregate Performance	
5 GFLOPS (peak)	50 GFLOPS (peak)
2,500 Mbytes memory	20,000 Mbytes memory
22,000 Mbytes disk space	22,000 Mbytes disk space
64,000 Mbytes of tape drive capacity	64,000 Mbytes of tape drive capacity
Constituent Parts	
256 processors (Weitek XL8032)	306 processors (two i860s each)
36 BSB crates	36 BSB crates
271 BSIBs	271 BSIBs
32 disk drives	32 disk drives
32 tape drives	32 tape drives

Canopy

A powerful software environment is an important and innovative part of the ACPMAPS system; it is called Canopy. Canopy is designed to support computing for grid oriented problems that map onto a lattice, defined generally as a set of sites, each of which has defined directions to a set of neighboring sites.

For example, a three dimensional grid is a lattice where the set of sites is the set of all points where the grid lines cross. The calculations performed by Fermilab and collaborating physicists make use of a four-dimensional space-time lattice to perform QCD calculations of the strong force. When running lattice gauge software under Canopy, ACPMAPS typically performs at the high level of about 30% of its peak GFLOPS rating. User programming is in C, with certain Canopy kernels and common computational kernels in assembler. The level of abstraction in Canopy remains high. The scientist uses subroutines to create a grid with known connectivity and to manipulate the fields that are situated on these grids. Then, by writing task routines to do operations on a site, the parallelism is automatically invoked. The user need not be aware of the specific machine details such as the number of nodes it has or how the sites are distributed among the nodes.

Summary

Motivated by an ongoing need to have cost effective computing, Fermilab has made some important contributions to distributed memory parallel computing in both hardware and software. By adopting the philosophy of incorporating technology at the highest commercially available level, these products will become even more robust.